Avoiding Communication at Exascale

James Demmel
demmel@cs.berkeley.edu
UC Berkeley, Lawrence Berkeley National Lab

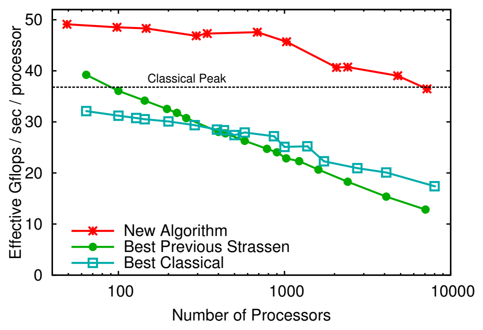
The cost of an algorithm, measured in time or energy, is a function of both computation, the number of arithmetic operations performed, and communication, the amount of data movement. Communication costs include both data movement between levels of the memory hierarchy and between processors over a network, and depend not just on the number of words sent, but the number of messages in which the words are packed, and the path taken through the network. Communication cost can be orders of magnitude larger than computation cost per operation on modern computer architectures, and the gap is only expected to widen in future systems, including exascale. This motivates us to design algorithms that minimize communication, and if possible to attain communication lower bounds. We call this a "communication-avoiding" (CA) approach to algorithmic design.

The biggest successes so far have been in dense linear algebra, sparse linear algebra and n-body problems, 3 of the "seven dwarfs" of computational science. We have also recently discovered a large generalization of these results, described below.

Our starting point was extending the well-known bound for dense matrix multiplication, to *all* direct linear algebra algorithms, for dense or sparse matrices, for sequential or parallel algorithms, and for homogeneous or heterogeneous architectures (this won the SIAM Linear Algebra Prize). It also applies to some graph algorithms like all-pairs-shortest-paths. Comparing to the existing algorithms in libraries like LAPACK and ScaLAPACK, we realized that most of them performed asymptotically more communication than required by the lower bound, leading us to begin inventing new algorithms for most direct linear algebra problems that are asymptotically faster than before. This includes matrix multiplication, tensor contractions, solving linear systems, least squares problems, eigenvalue problems and singular value problems.

We also developed the first communication lower bounds for Strassen's matrix multiplication algorithm, and the first communication-optimal implementation. Speedups and strong scalability on a Cray XT-4 are illustrated in the figure below, where this algorithm is the fastest known (the "best classical" is our own algorithm too).

We also generated performance models for an Exascale machine (using the 2018 Swimlane 1 extrapolation model), to predict speedups for these algorithms over conventional implementations. For example, up to 5.3x speedups were predicted for Strassen matrix multiplication over classical matrix multiplication, and for LU up to 4.5x speedups were predicted. The



latter speedup was due to a predicted 95% decrease in communication, leaving the problem compute-bound. Depending on the relative energy costs of communication and computation, the energy savings could possible be much larger.

For sparse, iterative linear algebra, which is based on Krylov subspace methods like CG, the cost of sparse-matrix-vector-multiplication (SpMV) and reductions for dot products are the communication bottlenecks. We have also developed new communication-optimal versions of these algorithms with large speedups, measured and predicted. For example, we have recently incorporated one of our methods into the bottom-solver of a production geometric multigrid code Boxlib, with up to 3x speedups in the bottom solve. For direct n-body interactions, we have also developed a new algorithm that up to 11.8x faster than predecessors.

Most recently, we have generalized these lower bounds and (some) optimal algorithms to any problem that can be expressed as nested loops accessing arrays whose subscripts are linear functions of the loop indices (including use of pointers, indirection, etc.). Finally, we have shown how to extend all these communication lower bounds to lower bounds on the energy needed to run the algorithm.

Because of all this progress, there are still a large number of open questions to answer: (1) We can potentially use our energy model to answer many questions, such as: What is the minimum energy required for a computation? Given a maximum allowed runtime T what is the minimum energy E needed to achieve it? Can we use our energy models and lower bounds for hardware-software codesign, to obtain provably energy-optimal solutions? (2) What is the most general set of preconditioners that we can use in our communication-avoiding Krylov methods? How do we best identify the most numerically stable version of these algorithms? How do we best incorporate these ideas into multigrid solvers? (3) How do we best avoid communication while taking advantage of "mathematical sparsity" that arises in matrix structures? This includes not just conventional sparse matrices, but low rank structures (eg hierarchical matrices), symmetries of the sort arising in tensor contractions in computational chemistry, and hybrids of these. (4) Based on the extension of our bounds and algorithms to general loop nests and array references, how many more algorithms (say of the "7 dwarfs" or their successors) can we optimize? Can this be automated using appropriate compiler technology?